Growing a Lab for Automated Upstream Testing: Challenges and Lessons Learned

Laura Nao, Collabora Ltd

Agenda

- Virtual tour of the Collabora LAVA Lab
- How does the Lab help with upstream testing
- Daily headaches and scaling challenges
- Maintenance and monitoring
- Performance tracking
- What's next

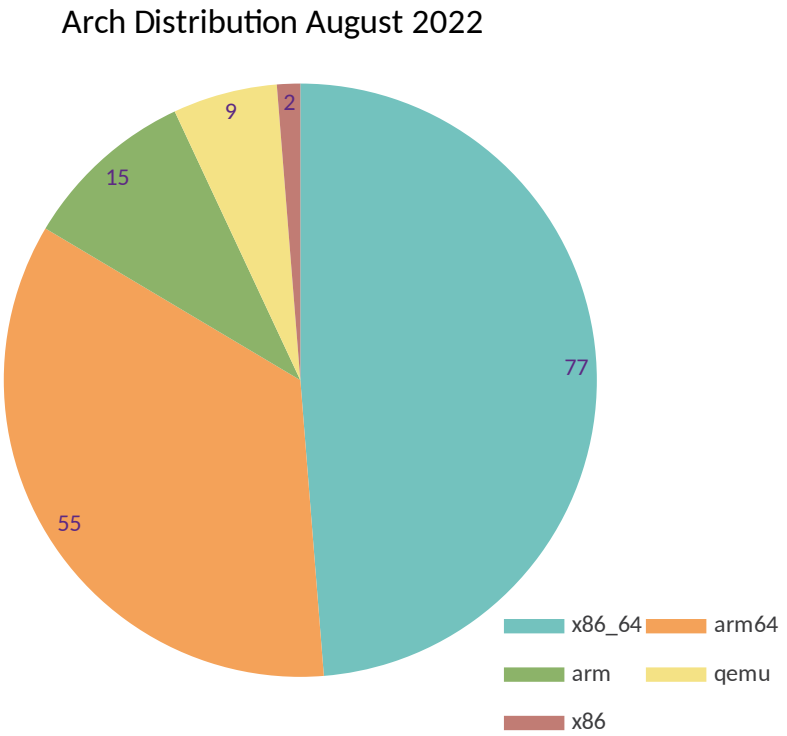
COLLABORA

Meet the Collabora LAVA Lab

- Lab for upstream testing maintained by Collabora
  - Helps to ensure that upstream projects remain functional when running on various hardware
- Runs LAVA (Linaro Automation and Validation Architecture) for functional testing on several platforms
- Located in Cambridge, UK

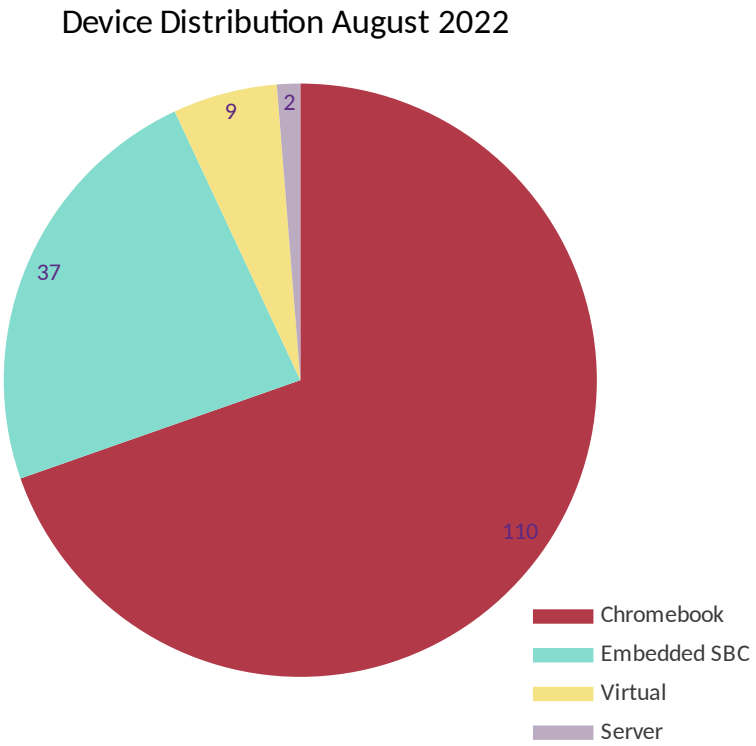
What's inside?

- 15 racks
- 158 devices of 32 different types
- 15 servers
  - 1 LAVA dispatcher per rack
- Network switches, debugging interfaces, USB hubs, power supplies, tons of cables, etc.

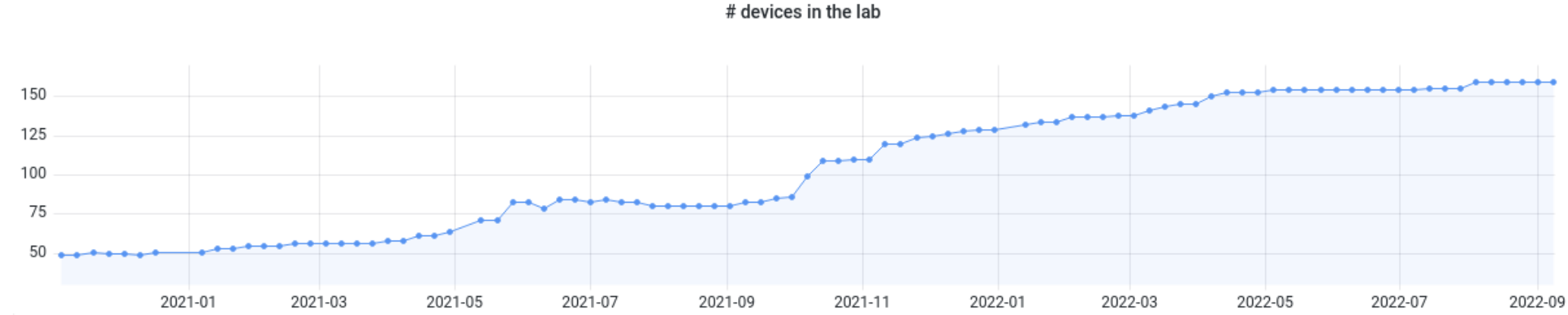


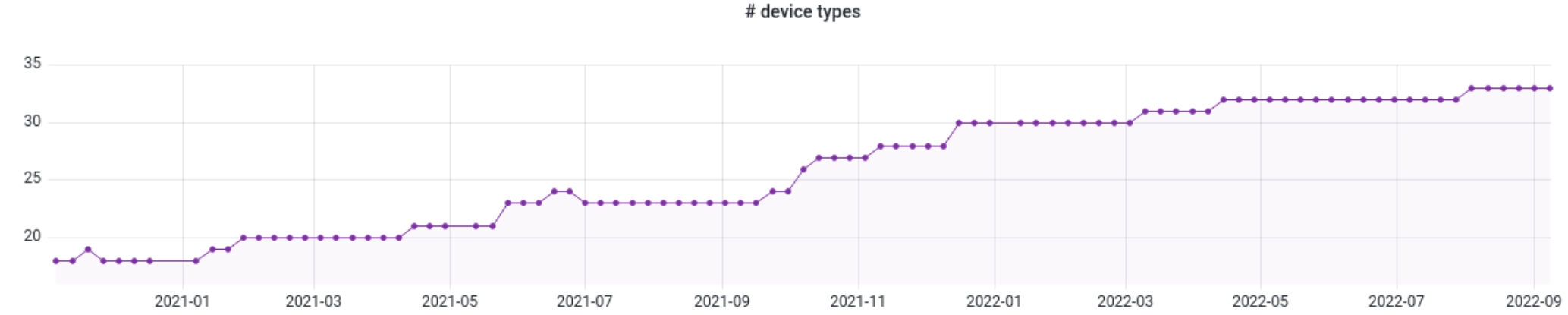
What's inside?

- 15 racks
- 158 devices of 32 different types
- 15 servers
  - 1 LAVA dispatcher per rack
- Network switches, debugging interfaces, USB hubs, power supplies, tons of cables, etc.



Lab Growth


devices in the lab
150
125
100
75
50
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07
2022-09


device types
35
30
25
20
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07
2022-09

In all its beauty:

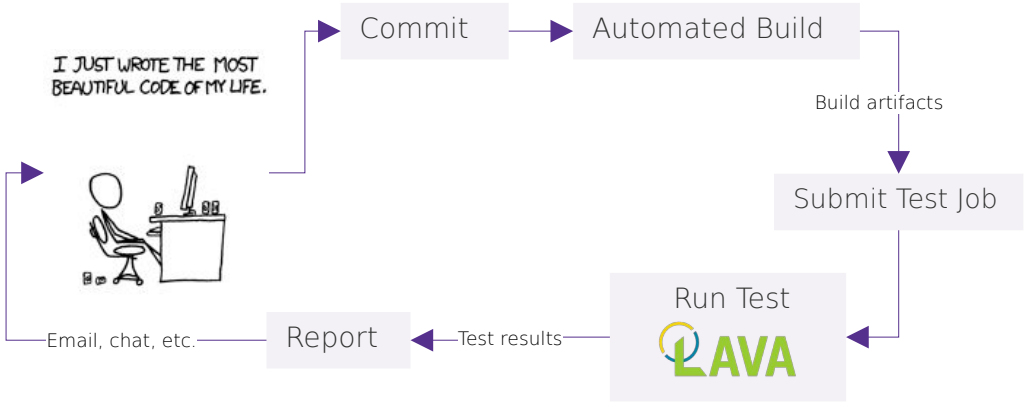
COLLABORA

LAVA

- LAVA is a CI system for deploying OSs onto physical and virtual hardware and running tests
  - Handles power control and console access
  - Schedules test jobs on DUTs
    - Boot testing
    - Bootloader testing
    - System level testing
- Designed for validation during development
  - Get feedback early and often
- Scalable scheduler
  - Can run thousands of tests across hundreds of devices on a single instance

Closing the CI Loop

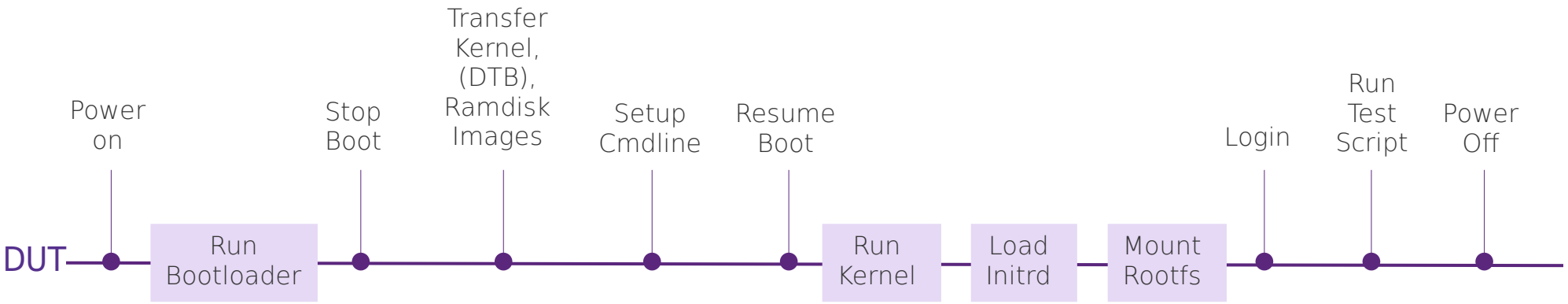
- Focused on kernel and system validation as part of a CI loop
  - Automates the deploy and boot phases
  - Test results can be formatted and exported in various formats
  - Building, test submission and feedback reporting is automated outside LAVA



Adding boards to LAVA

Base Requirements:

- Ability to be turned on/off remotely
- Ability to access a reliable console remotely
- Ability to boot arbitrary Kernel+(DTB)+System remotely



COLLABORA

Adding boards to LAVA

LAVA device configuration:

- Device Type Template
  - Template outlining the requirements to boot the device
    - e.g. What bootloader it runs, command line options for boot
- Device Dictionary
  - Template containing device-specific commands to interact with the device
    - e.g. Serial connection command, power on/off command, IP address
- Health Check
  - Checks the device can boot and deploy a test image + runs functional tests to verify the overall device status
    - e.g. Read battery percentage, check network connectivity, read temperature sensors

LAVA test job example (1/4)

Excerpt from KernelCI baseline test on a lazor Chromebook https://lava.collabora.dev/scheduler/job/7181364

```
device_type: sc7180-trogdor-lazor-limozeen
job_name: mainline-master-v6.0-rc3-363-g7726d4c3e60b-arm64-defconfig+arm64-chromebook-gcc-10-sc7180-trogdor-lazor-limozeen-baseline
priority: 45
timeouts:
  actions:
    power-off:
      seconds: 30
  job:
    minutes: 10
  queue:
    days: 2
visibility: public
context:
  extra_kernel_args: console_msg_format=syslog earlycon
```

LAVA test job example (2/4)

Excerpt from KernelCI baseline test on a lazor Chromebook https://lava.collabora.dev/scheduler/job/7181364

```
actions:
- deploy:
    dtb:
      url: http://storage.kernelci.org/mainline/master/v6.0-rc3-363-g7726d4c3e60b/arm64/defconfig+arm64-chromebook/gcc-10/dtbs/qcom/sc7180-trogdor-lazor-limozeen-nots-r5.dtb
    kernel:
      url: http://storage.kernelci.org/mainline/master/v6.0-rc3-363-g7726d4c3e60b/arm64/defconfig+arm64-chromebook/gcc-10/kernel/Image
    modules:
      compression: xz
      url: http://storage.kernelci.org/mainline/master/v6.0-rc3-363-g7726d4c3e60b/arm64/defconfig+arm64-chromebook/gcc-10/modules.tar.xz
    os: oe
    ramdisk:
      compression: gz
      url: http://storage.kernelci.org/images/rootfs/buildroot/buildroot-baseline/20220805.0/arm64/rootfs.cpio.gz
    timeout:
      minutes: 10
    to: tftp
```

LAVA test job example (3/4)

Excerpt from KernelCI baseline test on a lazor Chromebook https://lava.collabora.dev/scheduler/job/7181364

```
- boot:
    commands: ramdisk
    method: depthcharge
    prompts:
    - '/ #'
    timeout:
      minutes: 5
```

LAVA test job example (4/4)

Excerpt from KernelCI baseline test on a lazor Chromebook https://lava.collabora.dev/scheduler/job/7181364

```
- test:
    definitions:
    - from: inline
      lava-signal: kmsg
      name: dmesg
      path: inline/dmesg.yaml
      repository:
        metadata:
          description: baseline test plan
          environment:
          - lava-test-shell
          format: Lava-Test Test Definition 1.0
          name: baseline
          os:
          - debian
          scope:
          - functional
        run:
          steps:
          - KERNELCI_LAVA=y /bin/sh /opt/kernelci/dmesg.sh
    timeout:
      minutes: 1
```

Interacting with LAVA

- API - XML-RPC, REST
- lavacli
  - Python3 command line tool to push device type templates, dictionaries, submit jobs, etc.
  - Supports multiple identities to interact with multiple instances of LAVA
  - Uses the XML-RPC API
- LAVA Gitlab runner - https://gitlab.collabora.com/lava/lava-gitlab-runner
  - Bridges Gitlab to LAVA
  - Allows to submit a test job, monitor it and retrieve the raw log as a job artifact
  - Uses lava-api (relies on the LAVA REST API) - https://gitlab.collabora.com/lava/lava-api

COLLABORA

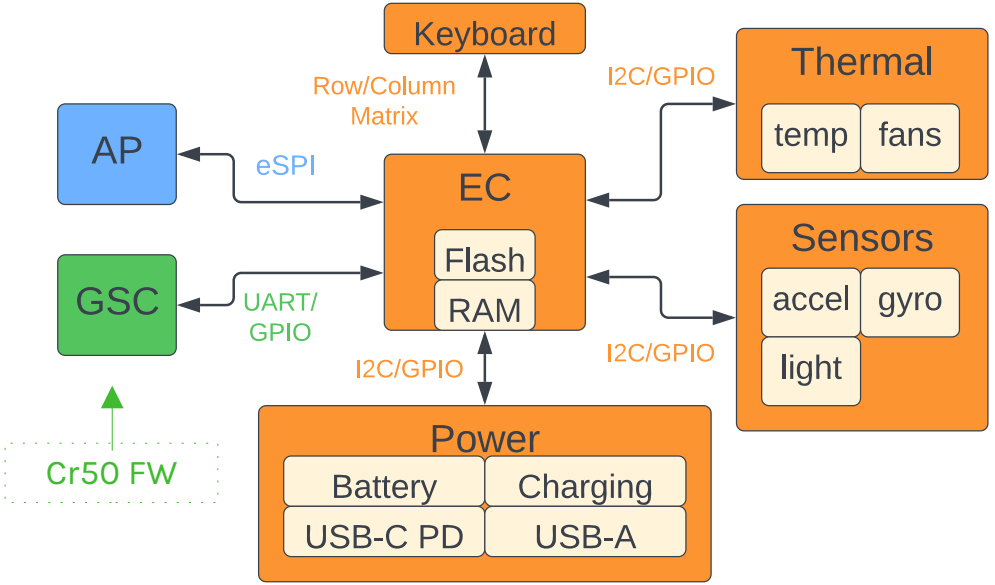
Adding a new device to the lab

- Device preparation
  - Setup the device for running tests
- LAVA device configuration
  - Device type template, dictionary, health check + network configuration
- Stress testing
  - Loop health check
- Device installation in the lab
- Upstream device type template

Chromebook Bring-Up in LAVA

Chromebook debugging:

- **CCD** (Closed Case Debugging):
  - Access relevant UARTs
  - Flash AP FW
  - Hold device in reset through GPIO
- **GSC** (Google Security Chip)
  - Runs Cr50 FW
    - Supports CCD



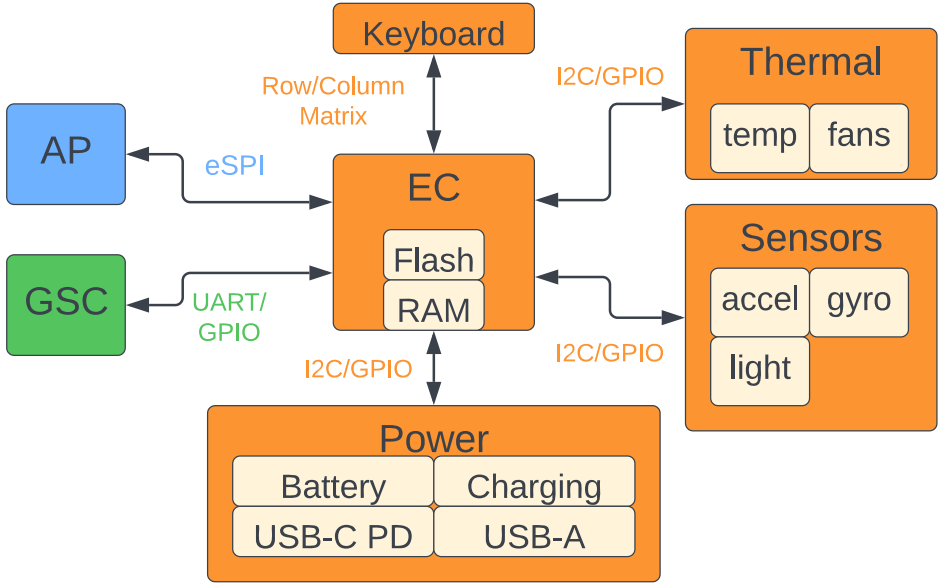
Pic: https://chromeos-dev.imgix.net/posts/embedded-controller/ec-detailed-block-diagram.svg

Chromebook Bring-Up in LAVA

Chromebook debugging:

- **Suzy-Q** – Interface with the GSC
  - Instruct Cr50 to enter debug mode
  - Expose Cr50 serial console

- **Hdctools** – Hardware Debug and Control Tools
  - SW tools to flash the device and control the DUT
  -



Pic: https://chromeos-dev.imgix.net/posts/embedded-controller/ec-detailed-block-diagram.svg

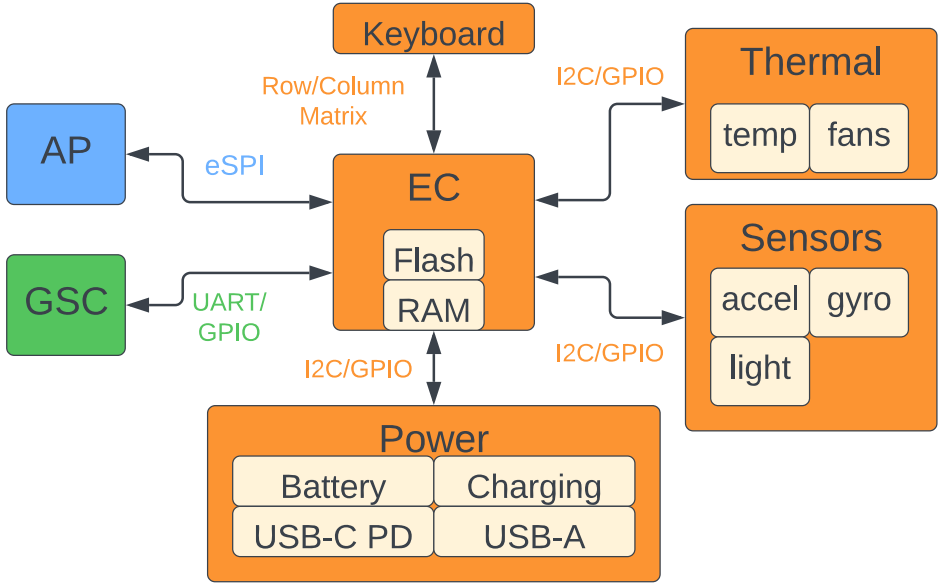
Chromebook Bring-Up in LAVA

Base Requirements:

- Ability to be turned on/off remotely – Suzy-Q + Hdctools
- Ability to access a reliable console remotely – Suzy-Q + Hdctools
- Ability to boot arbitrary Kernel+DTB+System remotely - ?

Chromebook Bring-Up in LAVA

- AP Boot Sequence:
  - Coreboot
    - Main system FW
  - Depthcharge
    - ChromeOS bootloader
  - ChromeOS

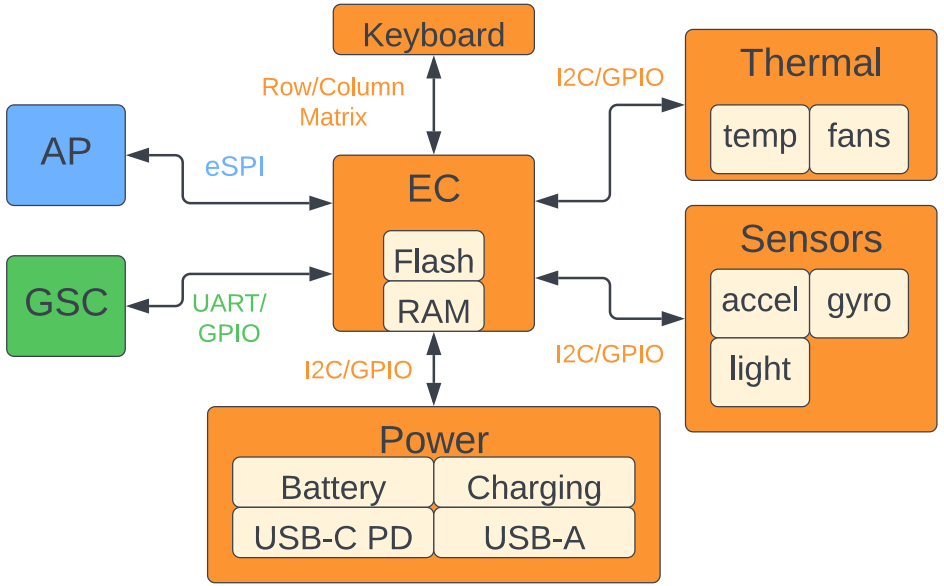


Pic: https://chromeos-dev.imgix.net/posts/embedded-controller/ec-detailed-block-diagram.svg

Chromebook Bring-Up in LAVA

- AP Boot Sequence:
  - Coreboot
  - Depthcharge
  - ~~ChromeOS~~
  - Kernel + DTB + Initrd

TFTP



Pic: https://chromeos-dev.imgix.net/posts/embedded-controller/ec-detailed-block-diagram.svg

Chromebook Bring-Up in LAVA

Minimum Requirements:

- Ability to be powered on/off remotely – CCD + Suzy-Q + Hdctools
- Ability to access text output remotely – CCD + Suzy-Q + Hdctools
- Ability to boot arbitrary Kernel+DTB+System remotely -

  Custom Deptcharge binary w/ TFTP and CLI support

COLLABORA

Staging vs Production

- staging.lava.collabora.dev
  - LAVA branch gets regularly rebased to stay close to upstream
  - LAVA patches get tested here
  - New devices are stress-tested here
  - Faulty devices are moved here for debugging
  - Helps discovering SW and HW issues early
- lava.collabora.dev
  - Production ready devices only
  - Device health continuously monitored

COLLABORA

Upstream CI Testing

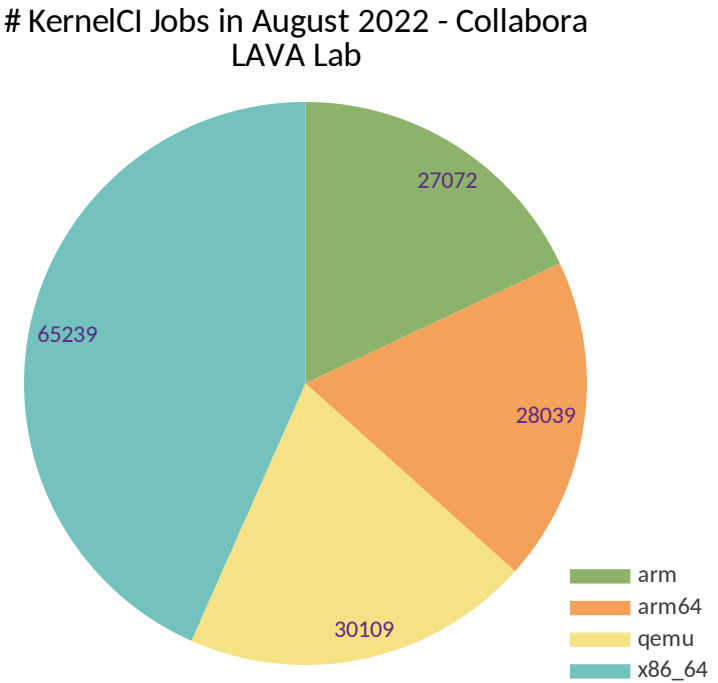
- Automated on-device testing is especially relevant for large scale open source projects
  - Tests code on a variety of different platforms in a standardized way
  - Helps find regressions and identify the root causes
  - Improves long-term maintenance and overall quality of the software
  - Helps catch mistakes early
- **KernelCI** and **Mesa's CI** have been leveraging the Collabora LAVA lab for their automated upstream testing and development workflows
  - More than a hundred thousand test jobs run every month

Upstream CI Testing

KernelCI

Mainline Linux Kernel continuous testing

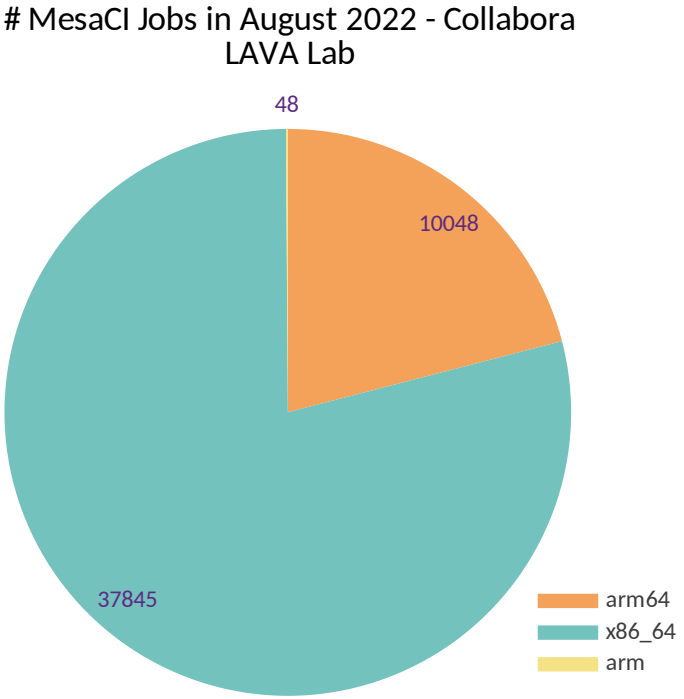
- Baseline tests
  - e.g. bootrr, dmesg
- Boot tests
  - e.g. boot-nfs, boot-fastboot
- Subsystem tests
  - e.g. igt, lc-compliance, ltp, v4l2-compliance
- Userspace tests
  - e.g. chromeos tast tests



Upstream CI Testing

Mesa CI

- Mesa pre-merge conformance testing and automated performance tracking
- APIs:
  - OpenGL, OpenGL ES, VA-API, Vulkan
- Drivers:
  - Iris, ANV, RadeonSI, RADV, Panfrost, Panvk, Freedreno, Turnip, LLVMPipe, Lavapipe, Softpipe, Etnaviv, Lima, v3d, v4c, Dozen, Virgl, Venus, Nouveau, Crocus
- Test suites:
  - dEQP, Khronos GL and VK CTS, Piglit, trace replaying for OpenGL, Vulkan and Direct3D, Skqp, va-utils



What could go wrong?

Common problems:

- HW degradation
  - e.g. Faulty cables, dead battery/power supply, dead SD card
- Network issues
  - e.g. Connectivity issues, IP address mismatches
- Rack setup issues
  - e.g. Disconnected cables, lid position, overheating
- FW bugs

Consequences:

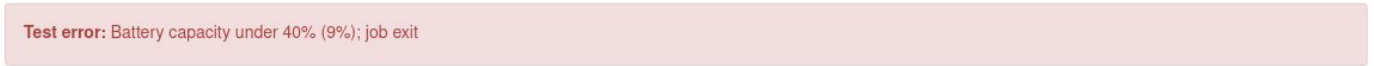
- Job queue growing
  - Canceled jobs
- Pipelines starving/failing
  - MRs blocked => angry user

What could go wrong?

Common issues should be either:

- Marked as `InfrastructureErrors` by LAVA

- Checked for regularly by health checks



COLLABORA

Best practices

- Monitor the device’s health
  - Ensure problems are addressed quickly + devices are replaced if needed
- Write robust health checks
  - Ensure faulty devices are taken down automatically
- Monitor LAVA InfrastructureError exceptions
  - Spot issues with specific racks or device types
- Device redundancy
  - Ensure good device coverage

Scaling Challenges

Scaling up means:

- More space required
- More HW equipment required
- More maintenance
- Increased load on LAVA's DB
  - Can impact the monitoring process

Performance Improvements

Recent achievements:

- LAVA performance tracking
  - Server statistics
  - DB statistics
    - Identify frequently made queries, track queries execution time and total time spent per query
  - Dummy load generator
    - Emulate DB high load scenario
- LAVA performance improvements
  - SQL optimizations
  - Branching logic optimizations
  - Pagination optimizations

COLLABORA

What's next?

- Keep adding new devices
  - Increase the lab capacity + cover variety of platforms from different vendors
- Keep improving our infrastructure and monitoring tools
- Keep reporting issues and sending patches to upstream LAVA
- Increase the coverage of test suites

Thank you!

We are hiring

col.la/careers